Validating Requirements for Fault Tolerant Systems Using Model Checking[1]

by Frank Schneider

A model checking analysis of a complex fault tolerant spacecraft controller was completed at the NASA/WVU IV&V Facility during 1996-1997, by Frank Schneider[2], Steve Easterbrook, and Jack Callahan, and Gerard Holzman[3]

The fault tolerant spacecraft controller analyzed was required to complete the execution of certain high priority tasks in the face of external spacecraft faults. The faults are characterized such that the controller could not proceed while a fault persisted. Accordingly, the controller had to stop execution of its high priority task; repair the fault; and subsequently continue with the execution of the high priority task . In order to make the process more efficient, and because certain subtasks should not be repeated each subtask was labeled with a "mark point" indicating that that subtask need not be repeated should the over-all task be interrupted. For example, a spacecraft turn with reaction wheels could not be repeated or the spacecraft heading would be incorrect. Or, if a soil sample were being gathered repeating a fetch for already retrieved tools would be redundant and wasteful of battery power. In this way all previously completed subtasks need not be repeated and only the last partially completed subtask has to be repeated. A typical sequence of events following a fault would be:

(1) spacecraft command detects the presence of an error caused by a fault elsewhere in the spacecraft,
(2) critical sequence execution is frozen,
(3) the fault causing the error is repaired,
(4) the critical sequence execution point is rolled back to the beginning of the partially completed sub-sequence (mark point), and
(5) re-execution of the high priority task is resumed.

To increase reliability and availability, a second redundant system controller is also executed in parallel with the main or prime controller. This backup controller could then take over control of the spacecraft bus should the prime system go down during the execution of these high priority tasks. A consequence of prime failure was that the recovery mechanism included the possibility that the prime system could fail while it was itself repairing an external fault.

Our work used model checking to validate the behavior of this system. The state space available to the entire system was several orders of magnitude too large to examine all possible behaviors. Accordingly, a model was abstracted from the system design that contained key elements that governed the system's behavior. By abstracting away detail not germane to the problem of interest, we were left with a partial specification. Our analysis focused on faults that occurred outside of the prime spacecraft controller. However, we were able to derive consequences to prime and backup spacecraft controllers failure with this approach.

Six separate requirements on the rollback scheme were validated. Each of the six requirements involved approximately 100, 000 states in the model, and took about 30 seconds each. The response and recovery in each case was to the injection of a single spacecraft fault in all possible ways, based on the model. Three anomalies were found in the system, showing that meaningful results could be derived from a modeling approach that uses only partial specifications. One anomaly was an error in the detailed requirements, and the other two were missing or ambiguous requirements.

Since this method allows the validation of partial specifications, we concluded that it is also an effective approach for use during the system development process. Here an effective approach is for an IV&V team to work in parallel with the development team. The IV&V team can then be responsible for making sure each incremental-partial-design element and its co-evolving implementation maintain fidelity with each other. Accordingly, the over-all development product should contain fewer errors. We expect this process to be especially useful, since it is based on design analysis and most errors in software development occur during the design phase or earlier.

We are currently pursuing this methodology on a spacecraft test system. Here the modeling approach will be used to check that the implementation conforms to the requirements by continuously monitoring the implementations' conformance to requirements whenever the test system is running.

This analysis was presented at the April 98 meeting of the Third Annual International Conference on Requirements Engineering (ICRE'98). It was named a "Best Paper" in its category " Safety, Survivability and Fault Tolerance." A copy of the full paper can be found at http://research.ivv.nasa.gov/ .

---

[1] The Research described in this article was carried out in part by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration, and in part by West Virginia University under NASA cooperative agreement #NCC 2-979, Reference herein to any specific commercial product, process, or service by trade, name , trademark manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government, the Jet Propulsion Laboratory, California Institute of Technology or West Virginia University

[2] Jet Propulsion Laboratory/California Institute of Technology MS 125-233 Pasadena, CA 91109.

[3] Computing Sciences Research, Bell Laboratories, Lucent Technologies, Murray Hill, NJ 07974